1- ¿Qué es un evento?
Un evento es una acción o suceso que ocurre en la página web y que puede ser detectado por JavaScript para ejecutar una respuesta, por ejemplo, hacer clic en un botón, mover el ratón, presionar una tecla, cargar la página, etc.

2- ¿Cuál es la sintaxis para manejar los eventos? Da un ejemplo.
La forma más recomendada es usando `addEventListener`:

```js
const boton = document.getElementById("miBoton");

boton.addEventListener("click", () => {
  alert("Hola desde addEventListener!");
});
```

3- ¿Consideras que es buena práctica utilizar el atributo onclick en el HTML, en lugar de usar addEventListener en el JS? Justifica.
No es buena práctica usar el atributo `onclick` directamente en el HTML porque, al hablar de organización y separación del código en sus respectivos archivos, HTML debería encargarse de la estructura, no de la lógica. Utilizar el atributo `onclick` es menos flexible, ya que este solo permite asignar una única función. Con `addEventListener` puedes agregar varias.
Además, se vuelve difícil de mantener. Si hay muchos eventos en el HTML, el código se vuelve menos limpio y más difícil de escalar.

4- Además del evento "click" utilizado en el taller, nombra otros 3 y explica cuándo los utilizarías.
`mouseover`: cuando el puntero entra sobre un elemento. Útil para menús o efectos hover dinámicos.

`keydown`: cuando el usuario presiona una tecla. Se puede usar para atajos de teclado o validaciones mientras se escribe.

`submit`: cuando se envía un formulario. Sirve para validar datos antes de enviarlos al servidor, por ejemplo.

5- Al dar click en el botón, sucedía que se disparaba el evento click del div también. ¿Cómo podríamos solucionar ese problema?
Esto ocurre debido a la propagación (o bubbling) de eventos. En el ejemplo del botón y el div, ambos elementos tienen un event listener para el evento click. Cuando se hace clic en el botón la alerta se ejecuta primero en el propio botón. Luego, el evento click burbujea hacia su elemento padre (div), que también tiene un event listener de click. Al llegar, ese listener se ejecuta, provocando que ambas alertas funcionen.

Este comportamiento se conoce como bubbling, porque el evento “sube” desde el elemento objetivo hacia sus elementos padres, ejecutando los listeners que coincidan en el viaje.

En realidad, todo el viaje del evento comienza desde el elemento raíz (document), pasa por la fase de captura, llega al elemento objetivo (el botón) y luego sube en la fase de burbujeo hasta llegar al document nuevamente, ejecutando los eventos que coincidan.

La manera para solucionar la propagación de eventos es utilizar `event.stopPropagation()`. Si se llama dentro de un listener, el evento no seguirá viajando hacia los elementos padres.

```js
document.getElementById('hijo').addEventListener('click', (event) => {
  console.log('Click en HIJO');
  event.stopPropagation(); // detiene el bubbling
});
```